Rewarding Probabilistic Hybrid Automata

Ernst Moritz Hahn

Institute of Software Chinese Academy of Sciences

September 27^{th}, 2013

joint work with Holger Hermanns

- real-world systems: combination of **digital** controller and **continuous** environment

- real-world systems: combination of **digital** controller and **continuous** environment
- often safeness critical

- real-world systems: combination of
  digital controller and
  continuous environment
- often safeness critical
- need to formally analyse such **hybrid** systems

Model Checking

"does a computing system fulfil its specification?"

Model Checking

"does a computing system fulfil its specification?"

- formal model $\mathcal{M}$ of system

Model Checking

"does a computing system fulfil its specification?"

- formal model $\mathcal{M}$ of system
- specification ϕ

Model Checking

"does a computing system fulfil its specification?"

- formal model $\mathcal{M}$ of system
- specification ϕ
- automatic proof or refutation of

$$\mathcal{M} \models \phi$$

Model Checking

"does a computing system fulfil its specification?"

- formal model $\mathcal{M}$ of system
- specification ϕ
- automatic proof or refutation of

$$\mathcal{M} \models \phi$$

- example: $\phi =$ temperature always below 37° celsius

Model Checking

"does a computing system fulfil its specification?"

- formal model $\mathcal{M}$ of system
- specification ϕ
- automatic proof or refutation of

$$\mathcal{M} \models \phi$$

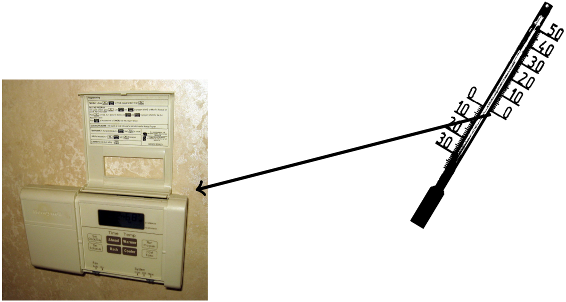
- example: $\phi =$ temperature always below 37° celsius

$\longrightarrow$ initial condition $\longrightarrow$ error

here: temperature equal to or above 37° celsius

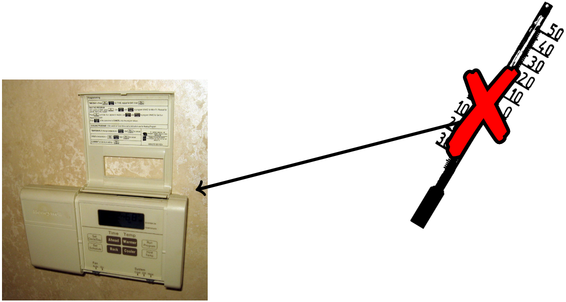
Probabilities

- probabilistic behaviour in system



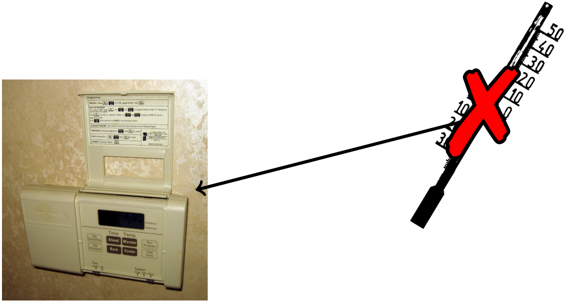
Probabilities

- probabilistic behaviour in system
  e.g. sensors might fail with given probability



Probabilities

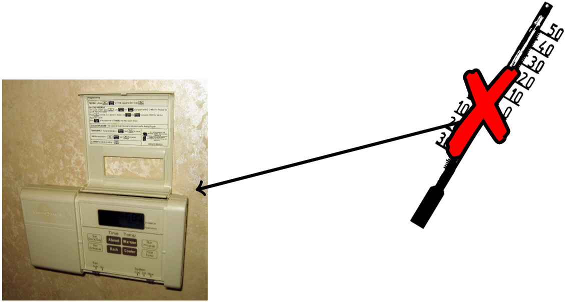
- probabilistic behaviour in system
  e.g. sensors might fail with given probability



- thus, cannot always show complete safeness

Probabilities

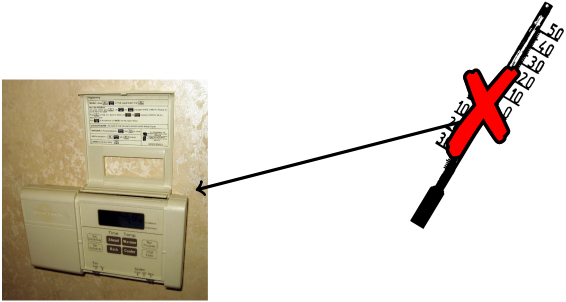
- probabilistic behaviour in system
  e.g. sensors might fail with given probability



- thus, cannot always show complete safeness
- want **quantitative** bounds on system behaviour
  e.g. “max prob to go above 37° within 20 years: $\leq 10^{-40}$”

Probabilities

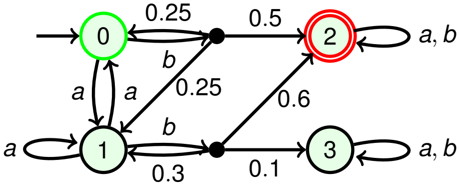
- probabilistic behaviour in system
  e.g. sensors might fail with given probability



- thus, cannot always show complete safeness
- want **quantitative** bounds on system behaviour
  e.g. "max prob to go above 37° within 20 years: $\leq 10^{-40}$"
- must integrate probabilistic behaviour in system model

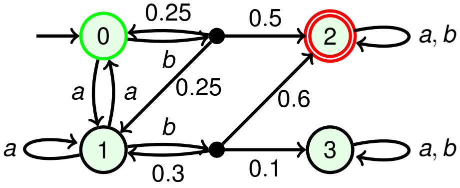
Analysis of Stochastic Hybrid Systems

- apply **model checking** to stochastic hybrid systems
  explore all states of the model
  combine with property
  apply analysis method to analyse state-transition system

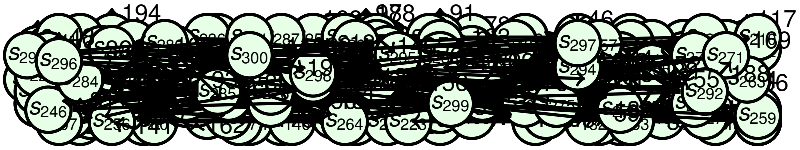


Analysis of Stochastic Hybrid Systems

- apply **model checking** to stochastic hybrid systems
  explore all states of the model
  combine with property
  apply analysis method to analyse state-transition system

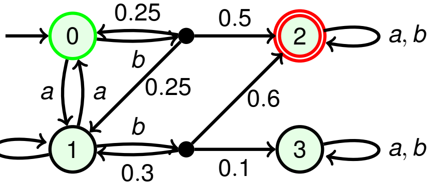


- problem: state space **uncountably large**



Analysis of Stochastic Hybrid Systems

- apply **model checking** to stochastic hybrid systems
  explore all states of the model
  combine with property
  apply analysis method to analyse state-transition system



- problem: state space **uncountably large**

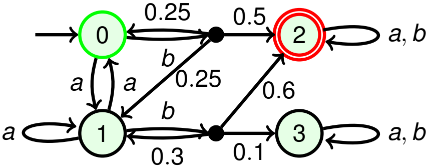
- thus, cannot be constructed explicitly

Abstraction of Stochastic Hybrid Systems

- idea: combine to finitely many **abstract** states
- apply model checking there

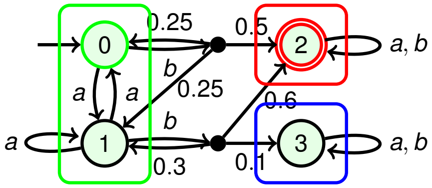
Abstraction of Stochastic Hybrid Systems

- idea: combine to finitely many **abstract** states
- apply model checking there



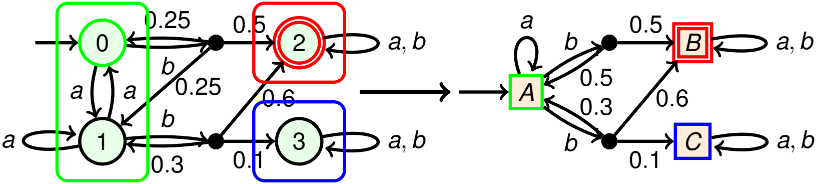
Abstraction of Stochastic Hybrid Systems

- idea: combine to finitely many **abstract** states
- apply model checking there



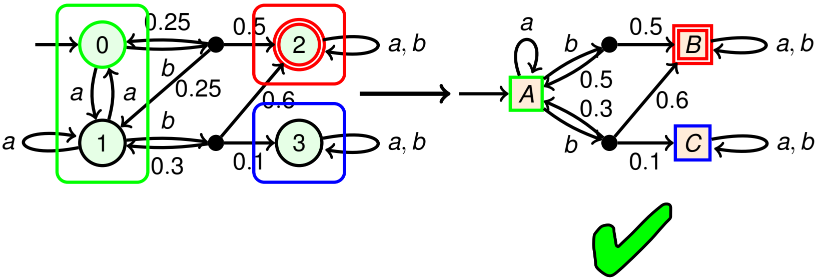
Abstraction of Stochastic Hybrid Systems

- idea: combine to finitely many **abstract** states
- apply model checking there



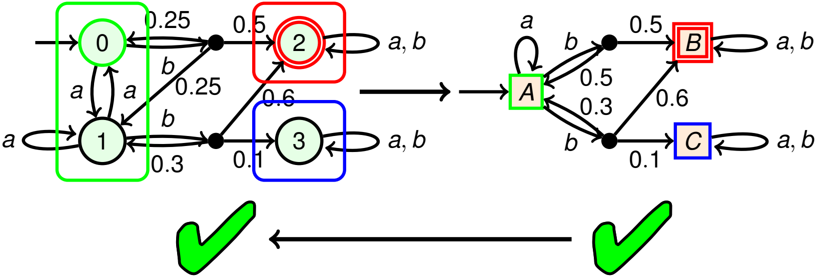
Abstraction of Stochastic Hybrid Systems

- idea: combine to finitely many **abstract** states
- apply model checking there



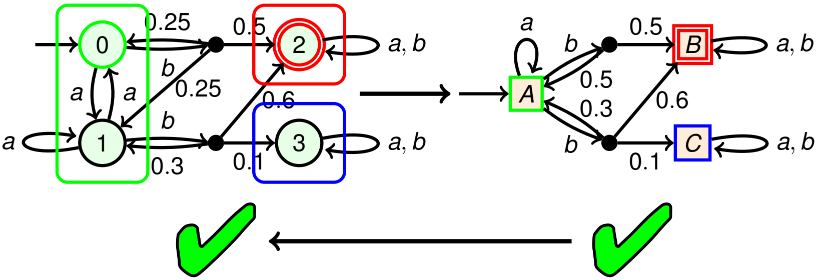
Abstraction of Stochastic Hybrid Systems

- idea: combine to finitely many **abstract** states
- apply model checking there



Abstraction of Stochastic Hybrid Systems

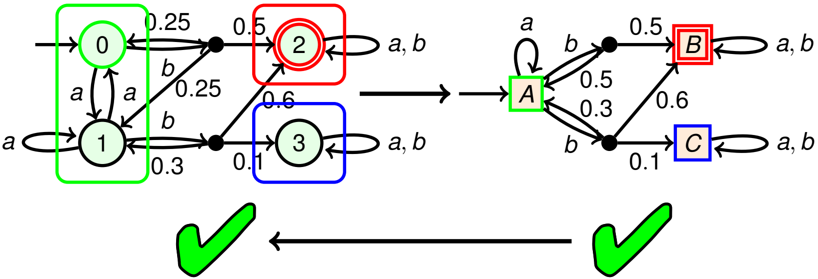
- idea: combine to finitely many **abstract** states
- apply model checking there



- how to construct such a finite model?

Abstraction of Stochastic Hybrid Systems

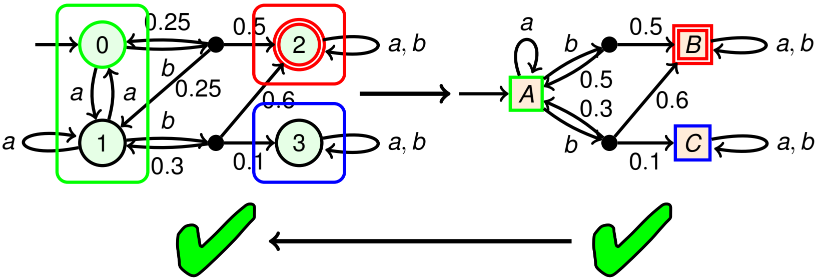
- idea: combine to finitely many **abstract** states
- apply model checking there



- how to construct such a finite model?
- correctness?

Abstraction of Stochastic Hybrid Systems

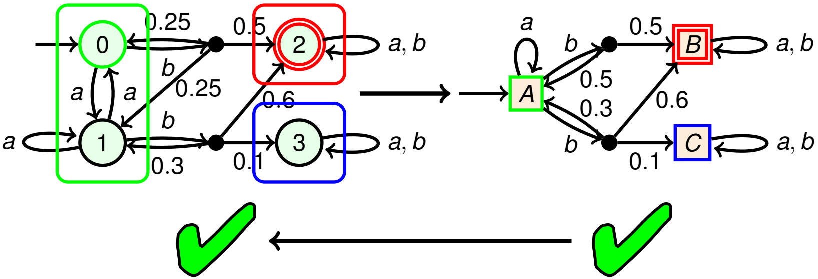
- idea: combine to finitely many **abstract** states
- apply model checking there



- how to construct such a finite model?
- correctness?
- which models can we handle?

Abstraction of Stochastic Hybrid Systems

- idea: combine to finitely many **abstract** states
- apply model checking there



- how to construct such a finite model?
- correctness?
- which models can we handle?
- and which properties?

Contribution

- generic framework for general stochastic hybrid systems

Contribution

- generic framework for general stochastic hybrid systems
- provides conservative bounds for properties

Contribution

- generic framework for general stochastic hybrid systems
- provides conservative bounds for properties
- requires no manual intervention

Contribution

- generic framework for general stochastic hybrid systems
- provides conservative bounds for properties
- requires no manual intervention

classical hybrid automata

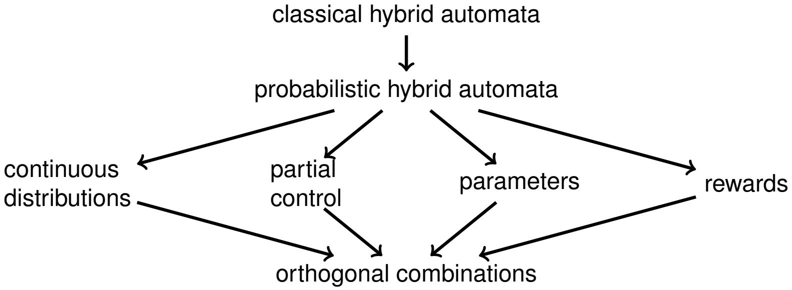
$\downarrow$

probabilistic hybrid automata

- builds on classical hybrid solvers (important research area)

Contribution

- generic framework for general stochastic hybrid systems
- provides conservative bounds for properties
- requires no manual intervention



- builds on classical hybrid solvers (important research area)
- applicable to wide area of models and properties

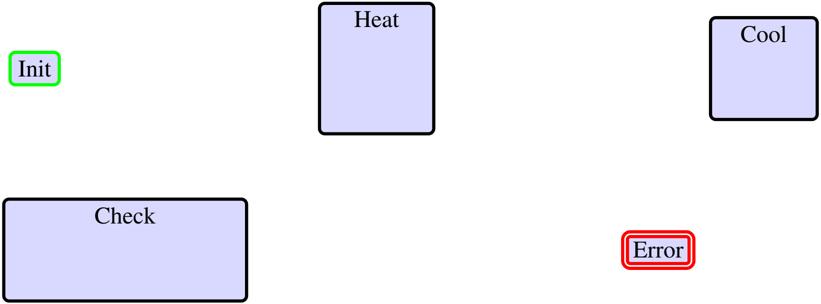
Classical Hybrid Automata

Hybrid Automata (HA)

$\mathcal{H} = (M, \overline{m} \qquad\qquad\qquad\qquad)$

[MalerMP91]

- M: finite set of **modes**
- $\overline{m}$: **initial mode**

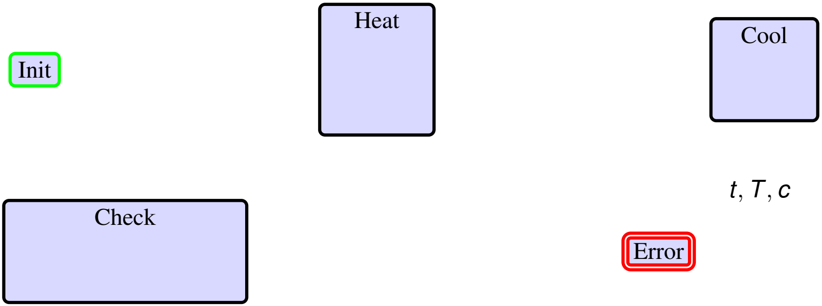


Hybrid Automata (HA)

$\mathcal{H} = (M, \overline{m}, k \qquad\qquad\qquad\qquad)$

[MalerMP91]

- M: finite set of **modes**
- $\overline{m}$: **initial mode**
- k: **dimension** of the automaton

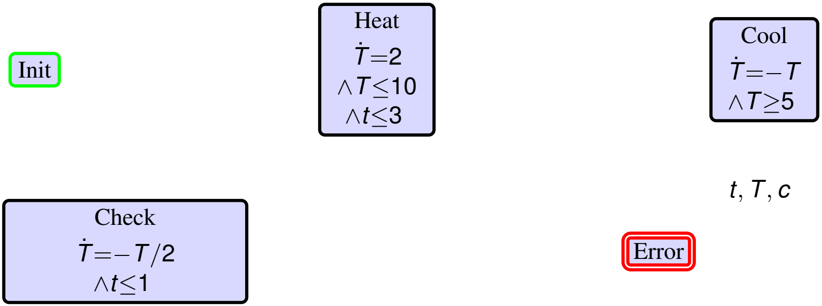


Hybrid Automata (HA)

$\mathcal{H} = (M, \overline{m}, k, \langle Post_m \rangle_{m \in M} \quad)$

[MalerMP91]

- M: finite set of **modes**
- $\overline{m}$: **initial mode**
- k: **dimension** of the automaton
- $Post_m$: timed behaviour



Hybrid Automata (HA)

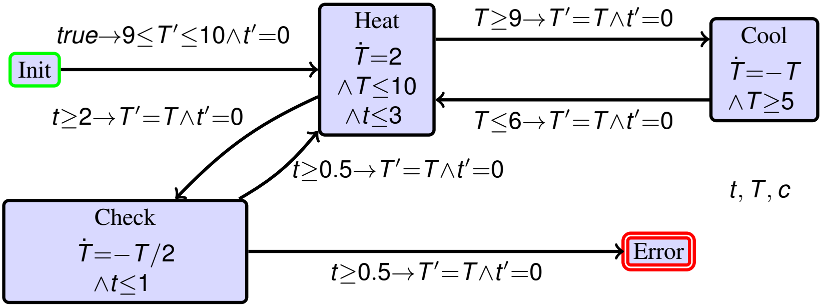
$\mathcal{H} = (M, \overline{m}, k, \langle Post_m \rangle_{m \in M}, Cmds)$

[MalerMP91]

- M: finite set of **modes**
- $\overline{m}$: **initial mode**
- k: **dimension** of the automaton
- $Post_m$: timed behaviour
- $Cmds$: finite set of **guarded commands** $g \to u$

  g: **guard**

  u: **update function**

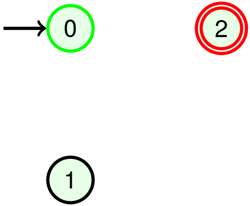


Labelled Transition Systems (LTS)

$\mathcal{M} = (S, \overline{s} \qquad\quad)$

[Keller76]

- S: set of **states**
- $\overline{s}$: **initial state**

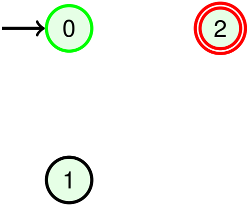


Labelled Transition Systems (LTS)

$\mathcal{M} = (S, \overline{s}, Act \quad)$

[Keller76]

- S: set of **states**
- $\overline{s}$: **initial state**
- Act: **actions**

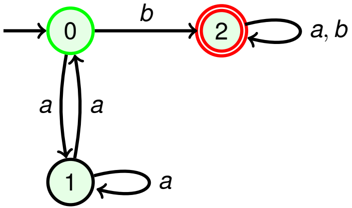


Labelled Transition Systems (LTS)

$\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$

[Keller76]

- S: set of **states**
- $\overline{s}$: **initial state**
- Act: **actions**
- $\mathcal{T}$: **transition matrix**

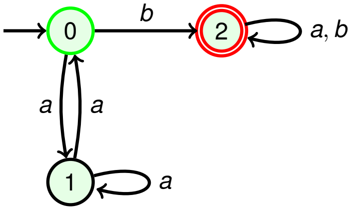


Labelled Transition Systems (LTS)

$\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$

[Keller76]

- S: set of **states**
- $\overline{s}$: **initial state**
- Act: **actions**
- $\mathcal{T}$: **transition matrix**

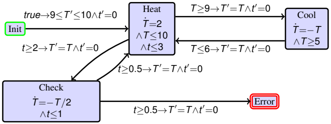


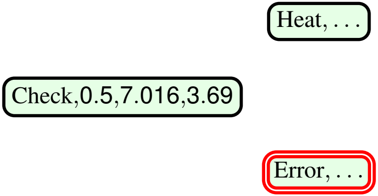
- **path**: state-action sequence legal by $\mathcal{T}$
  e.g. $0 \rightarrow a \rightarrow 1 \rightarrow a \rightarrow 1 \rightarrow a \rightarrow 0 \rightarrow b \rightarrow 2 \rightarrow \ldots$

Semantics of Hybrid Automata

LTS $[\![\mathcal{H}]\!] = (S, \overline{s} \qquad\qquad)$

- $S = M \times \mathbb{R}^k$
- $\overline{s} = (\overline{m}, 0, \ldots, 0)$

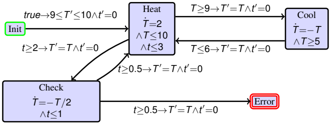




Semantics of Hybrid Automata

LTS $[\![\mathcal{H}]\!] = (S, \overline{s}, Act \quad)$

- $S = M \times \mathbb{R}^k$
- $\overline{s} = (\overline{m}, 0, \ldots, 0)$
- $Act = \mathbb{R}_{\geq 0} \uplus Cmds$



Heat, . . .

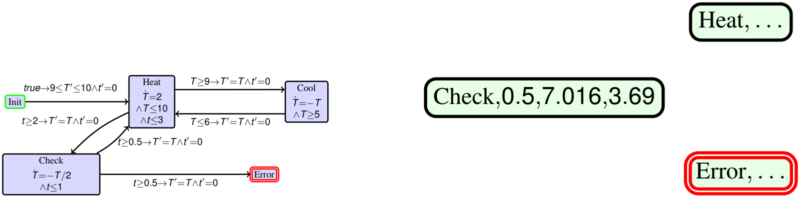
Check,0.5,7.016,3.69

Error, . . .

Semantics of Hybrid Automata

LTS $[\![\mathcal{H}]\!] = (S, \overline{s}, Act, \mathcal{T})$

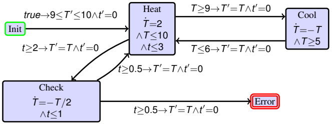
- $S = M \times \mathbb{R}^k$
- $\overline{s} = (\overline{m}, 0, \ldots, 0)$
- $Act = \mathbb{R}_{\geq 0} \uplus Cmds$
- $\mathcal{T}$: for $s \in S$ have transitions

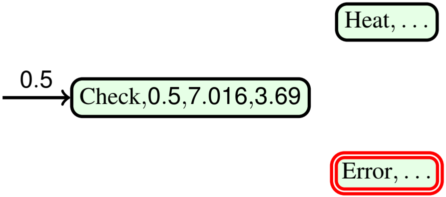


Semantics of Hybrid Automata

LTS $[\![\mathcal{H}]\!] = (S, \overline{s}, Act, \mathcal{T})$

- $S = M \times \mathbb{R}^k$
- $\overline{s} = (\overline{m}, 0, \ldots, 0)$
- $Act = \mathbb{R}_{\geq 0} \uplus Cmds$
- $\mathcal{T}$: for $s \in S$ have transitions from time t by $Post_m(s, \mathrm{t})$

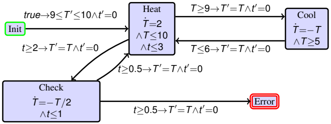


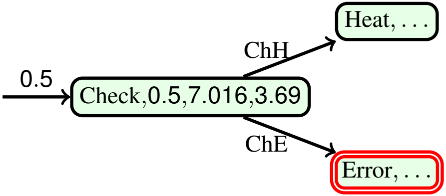


Semantics of Hybrid Automata

LTS $[\![\mathcal{H}]\!] = (S, \overline{s}, Act, \mathcal{T})$

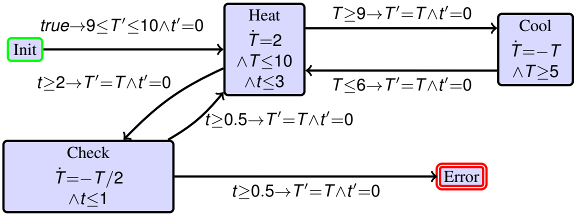
- $S = M \times \mathbb{R}^k$
- $\overline{s} = (\overline{m}, 0, \ldots, 0)$
- $Act = \mathbb{R}_{\geq 0} \uplus Cmds$
- $\mathcal{T}$: for $s \in S$ have transitions
  from time t by $Post_m(s, \mathrm{t})$
  from command $g \to u$ by $u(s)$ if g fulfilled





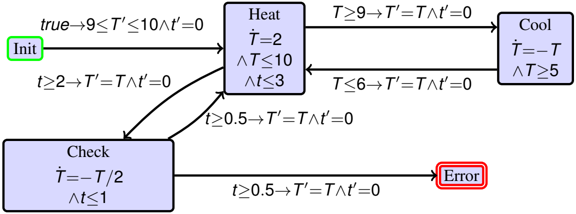
Reachability

Given $\mathcal{H}$, does there exist a path reaching an unsafe mode?



Reachability

Given $\mathcal{H}$, does there exist a path reaching an unsafe mode?

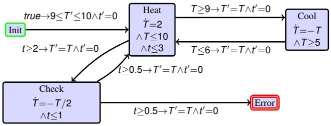


$(\text{Init}, 0, 0, 0) \rightarrow \text{IH} \rightarrow (\text{Heat}, 0, 9, 0) \rightarrow 0.5 \rightarrow (\text{Heat}, 0.5, 10, 0.5) \rightarrow \text{HCo} \rightarrow (\text{Cool}, 0, 10, 0.5) \rightarrow 0.69 \rightarrow (\text{Cool}, 0.69, 5.016, 1.19) \rightarrow \text{CoH} \rightarrow (\text{Heat}, 0, 5.016, 1.19) \rightarrow 2 \rightarrow (\text{Heat}, 2, 9.016, 3.19) \rightarrow \text{HCh} \rightarrow (\text{Check}, 0, 9.016, 3.19) \rightarrow 0.5 \rightarrow (\text{Check}, 0.5, 7.016, 3.69) \rightarrow \text{ChE} \rightarrow (\text{Error}, 0, 7.016, 3.69)$

Abstraction of Hybrid Automata

LTS $\mathcal{M} = (\mathbf{A}, \bar{\mathbf{z}} \qquad\qquad\qquad)$

- **A**: **covering**
- $\bar{\mathbf{z}}$: contains initial state



$\mathbf{z}_0$ Init

$\mathbf{z}_1$ Heat: $t{\geq}0, c{\geq}0, t{\leq}c, T{\leq}10$

$\mathbf{z}_2$ Check: $t{\geq}0, c{\geq}2, t{\leq}c{-}2, T{\leq}10$

$\mathbf{z}_3$ Error: $c{\leq}5$

$\mathbf{z}_4$ Cool: $t{\geq}0, c{\geq}0, t{\leq}c, T{\leq}10$

$\mathbf{z}_5$ Heat: $t{\geq}0, c{\geq}2.5, t{\leq}c{-}2.5, T{\leq}10$

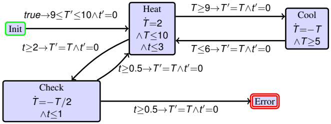
$\mathbf{z}_6$ Check: $t{\geq}0, c{\geq}4.5, t{\leq}c{-}4.5, T{\leq}10$

$\mathbf{z}_7$ Heat: $t{\geq}0, c{\geq}0, t{\leq}c{-}5, T{\leq}10$

Abstraction of Hybrid Automata

LTS $\mathcal{M} = (\mathbf{A}, \overline{\mathbf{z}}, \{\tau\} \uplus \mathit{Cmds} \qquad)$

- **A: covering**
- $\overline{\mathbf{z}}$: contains initial state
- τ: abstract timed action
- *Cmds*: commands of $\mathcal{H}$



$\mathbf{z}_0$ Init

$\mathbf{z}_1$ Heat: $t \geq 0, c \geq 0, t \leq c, T \leq 10$

$\mathbf{z}_2$ Check: $t \geq 0, c \geq 2, t \leq c-2, T \leq 10$

$\mathbf{z}_3$ Error: $c \leq 5$

$\mathbf{z}_4$ Cool: $t \geq 0, c \geq 0, t \leq c, T \leq 10$

$\mathbf{z}_5$ Heat: $t \geq 0, c \geq 2.5, t \leq c-2.5, T \leq 10$

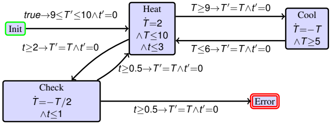
$\mathbf{z}_6$ Check: $t \geq 0, c \geq 4.5, t \leq c-4.5, T \leq 10$

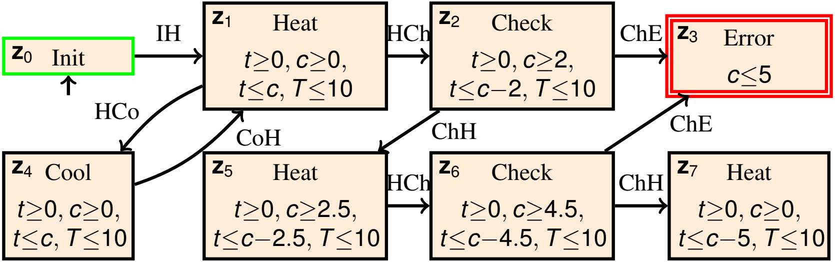
$\mathbf{z}_7$ Heat: $t \geq 0, c \geq 0, t \leq c-5, T \leq 10$

Abstraction of Hybrid Automata

LTS $\mathcal{M} = (\mathbf{A}, \bar{\mathbf{z}}, \{\tau\} \uplus \mathit{Cmds}, \mathcal{T}_{\text{abs}})$

- **A: covering**
- $\bar{\mathbf{z}}$: contains initial state
- τ: abstract timed action
- *Cmds*: commands of $\mathcal{H}$
- $\mathcal{T}_{\text{abs}}$: transfer transitions to abstraction

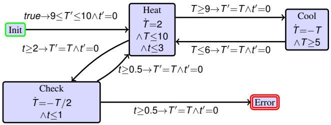


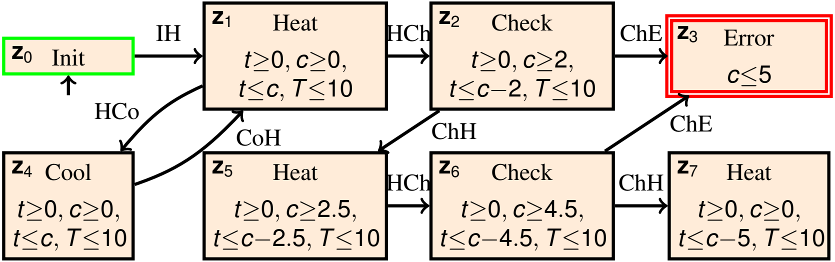


Abstraction of Hybrid Automata

LTS $\mathcal{M} = (\mathbf{A}, \bar{\mathbf{z}}, \{\tau\} \uplus \mathit{Cmds}, \mathcal{T}_{\text{abs}})$

- **A: covering**
- $\bar{\mathbf{z}}$: contains initial state
- τ: abstract timed action
- *Cmds*: commands of $\mathcal{H}$
- $\mathcal{T}_{\text{abs}}$: transfer transitions to abstraction





- wide tool support exists (HSolver, PHAVer, SpaceEx, etc.)

Correctness of HA Abstraction

- follows from **simulation relation** semantics $\rightarrow$ abstraction [Milner71]

Correctness of HA Abstraction

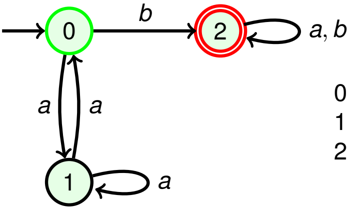
- follows from **simulation relation** semantics $\rightarrow$ abstraction
- $R \subseteq S \times \mathbf{A}$ [Milner71]

Correctness of HA Abstraction

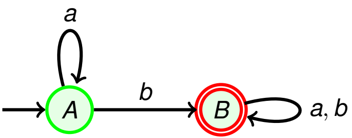
- follows from **simulation relation** semantics $\rightarrow$ abstraction
- $R \subseteq S \times \mathbf{A}$ [Milner71]
- $\overline{s}\, R\, \overline{\mathbf{z}}$

Correctness of HA Abstraction

- follows from **simulation relation** semantics $\rightarrow$ abstraction
- $R \subseteq S \times \mathbf{A}$ [Milner71]
- $\overline{s} \, R \, \overline{\mathbf{z}}$
- if $s \, R \, \mathbf{z}$,
  for a-labelled successor in simulated model,
  have a-labelled successor in simulating model,
  so that the two are also related

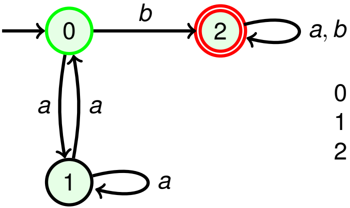


$0 \; R \; A$
$1 \; R \; A$
$2 \; R \; B$

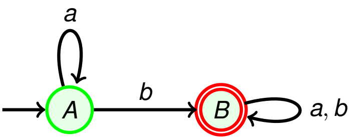


Correctness of HA Abstraction

- follows from **simulation relation** semantics $\rightarrow$ abstraction
- $R \subseteq S \times \mathbf{A}$ [Milner71]
- $\overline{s}\, R\, \overline{\mathbf{z}}$
- if $s\, R\, \mathbf{z}$,
  for a-labelled successor in simulated model,
  have a-labelled successor in simulating model,
  so that the two are also related



$0\ R\ A$
$1\ R\ A$
$2\ R\ B$



- maintains safeness

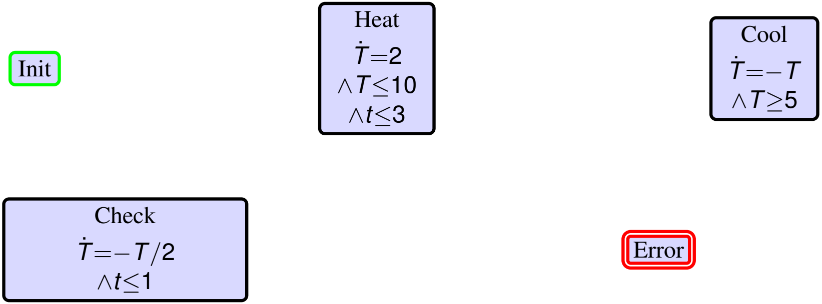
Analysis of Stochastic Hybrid Systems

Probabilistic Hybrid Automata (PHA)

$\mathcal{H} = (M, \overline{m}, k, \langle Post_m \rangle_{m \in M} \qquad)$

- M: finite set of modes
- $\overline{m}$: initial mode
- k: dimension of the automaton
- $Post_m$: timed behaviour

[Sproston00]



Probabilistic Hybrid Automata (PHA)

$\mathcal{H} = (M, \overline{m}, k, \langle Post_m \rangle_{m \in M}, Cmds)$

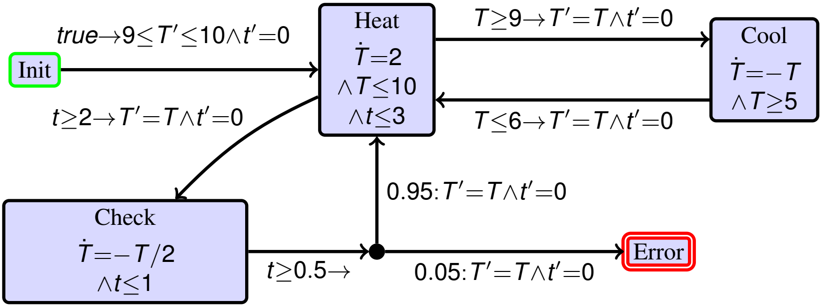
- M: finite set of modes
- $\overline{m}$: initial mode
- k: dimension of the automaton
- $Post_m$: timed behaviour
- $Cmds$: finite set of **probabilistic** guarded commands

[Sproston00]

$$g \to p_1{:}u_1 + \ldots + u_n{:}u_n$$

g: guard

u_i: updates p_i: **probabilities**

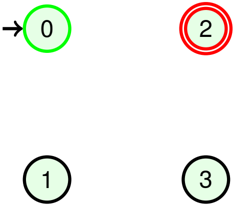


Probabilistic Automata (PA)

$\mathcal{M} = (S, \overline{s}, Act \quad)$

- S: set of states
- $\overline{s}$: initial state
- Act: actions

[SegalaL95]

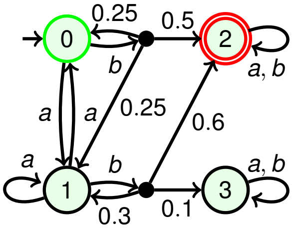


Probabilistic Automata (PA)

$\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$

[SegalaL95]

- S: set of states
- $\overline{s}$: initial state
- Act: actions
- $\mathcal{T}$: **probabilistic** transition matrix

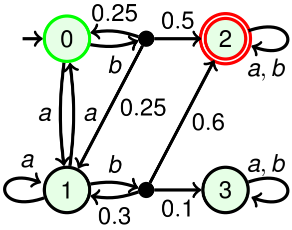


Probabilistic Automata (PA)

$\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$

[SegalaL95]

- S: set of states
- $\overline{s}$: initial state
- Act: actions
- $\mathcal{T}$: **probabilistic** transition matrix



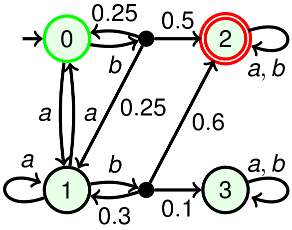
- **path**: sequence state-action-**distribution** e.g.
  $0 \rightarrow b \rightarrow [0 \mapsto 0.25, 1 \mapsto 0.25, 2 \mapsto 0.5] \rightarrow 2 \rightarrow a \rightarrow [2 \mapsto 1] \ldots$

Probabilistic Automata (PA)

$\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$

[SegalaL95]

- S: set of states
- $\overline{s}$: initial state
- Act: actions
- $\mathcal{T}$: **probabilistic** transition matrix



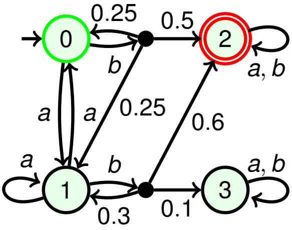
- **path**: sequence state-action-**distribution** e.g.
  $0 \to b \to [0 \mapsto 0.25, 1 \mapsto 0.25, 2 \mapsto 0.5] \to 2 \to a \to [2 \mapsto 1] \ldots$
- **scheduler** $\sigma \in Sched_{\mathcal{M}}$: fixes decisions over successors

Probabilistic Automata (PA)

$\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$

[SegalaL95]

- S: set of states
- $\overline{s}$: initial state
- Act: actions
- $\mathcal{T}$: **probabilistic** transition matrix

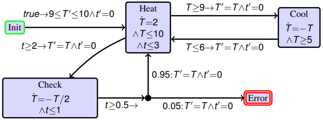


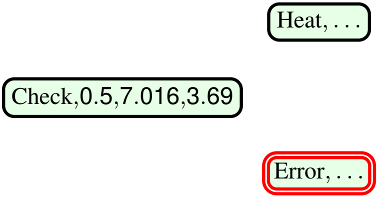
- **path**: sequence state-action-**distribution** e.g.
  $0 \rightarrow b \rightarrow [0{\mapsto}0.25, 1{\mapsto}0.25, 2{\mapsto}0.5] \rightarrow 2 \rightarrow a \rightarrow [2{\mapsto}1] \ldots$
- **scheduler** $\sigma \in Sched_{\mathcal{M}}$: fixes decisions over successors
- induces measure $Pr_{\mathcal{M},\sigma}$ on sets of paths

Semantics of Probabilistic Hybrid Automata

LTS $[\![\mathcal{H}]\!] = (S, \overline{s}, Act \quad)$

- $S = M \times \mathbb{R}^k$
- $\overline{s} = (\overline{m}, 0, \ldots, 0)$
- $Act = \mathbb{R}_{\geq 0} \uplus Cmds$

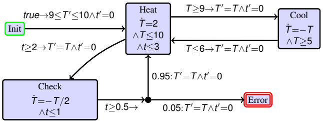


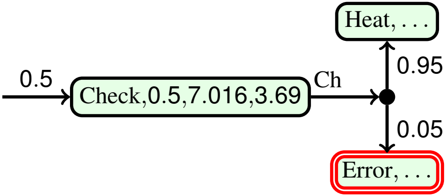


Semantics of Probabilistic Hybrid Automata

LTS $[\![\mathcal{H}]\!] = (S, \overline{s}, Act, \mathcal{T})$

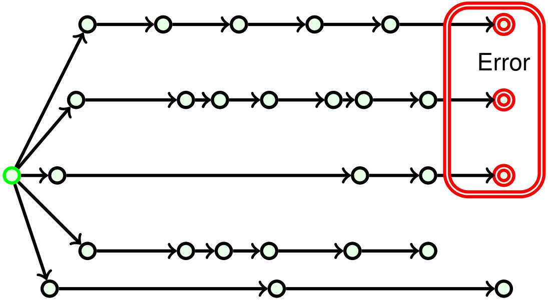
- $S = M \times \mathbb{R}^k$
- $\overline{s} = (\overline{m}, 0, \ldots, 0)$
- $Act = \mathbb{R}_{\geq 0} \uplus Cmds$
- $\mathcal{T}$: for $s = \in S$ have transitions
  from command $g \to p_1{:}u_1 + \ldots + u_n{:}u_n$ by $u(s)$ if g fulfilled
  from time t by $Post_m(s, \mathrm{t})$





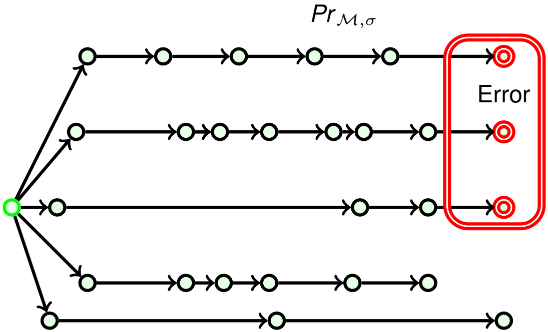
Probabilistic Reachability

- consideration of single path insufficient



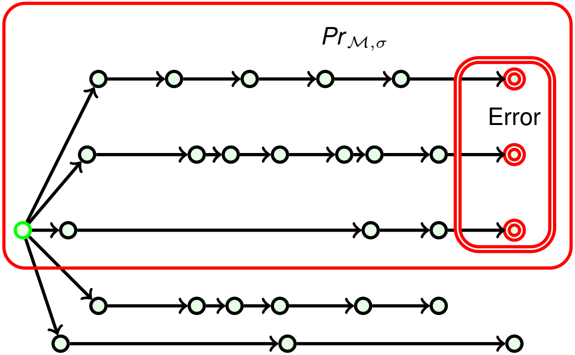
Probabilistic Reachability

- consideration of single path insufficient
- scheduler $\sigma \in \mathit{Sched}_{\mathcal{M}}$ induces measure on path sets



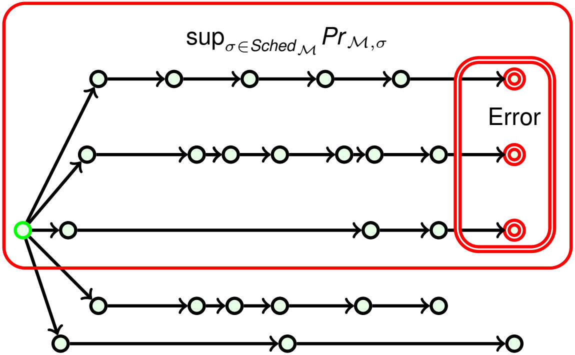
Probabilistic Reachability

- consideration of single path insufficient
- scheduler $\sigma \in Sched_{\mathcal{M}}$ induces measure on path sets
- want probability of paths reaching bad state

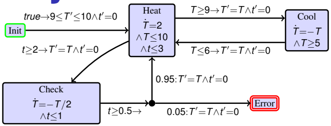


Probabilistic Reachability

- consideration of single path insufficient
- scheduler $\sigma \in \mathit{Sched}_{\mathcal{M}}$ induces measure on path sets
- want probability of paths reaching bad state
- interested in worst case, thus supremum over schedulers

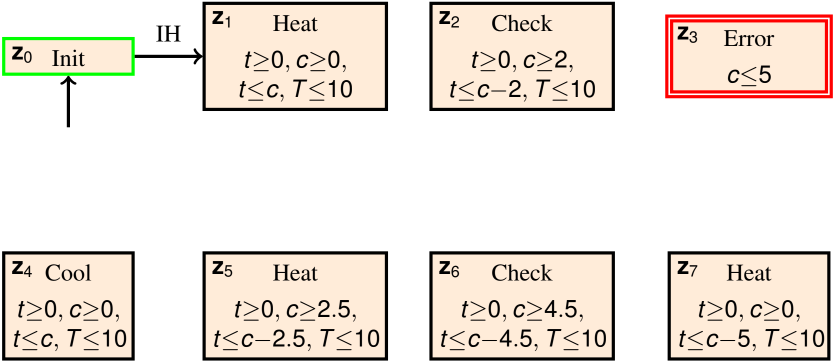


Abstraction of Probabilistic Hybrid Automata



PA $\mathcal{M} = (\mathbf{A}, \bar{\mathbf{z}}, \{\tau\} \uplus \mathit{Cmds}, \mathcal{T}_{\mathrm{abs}})$

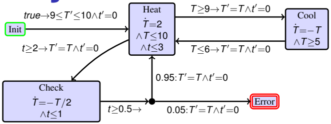
- $\mathbf{A}, \bar{\mathbf{z}}, \tau, \mathit{Cmds}$: as before

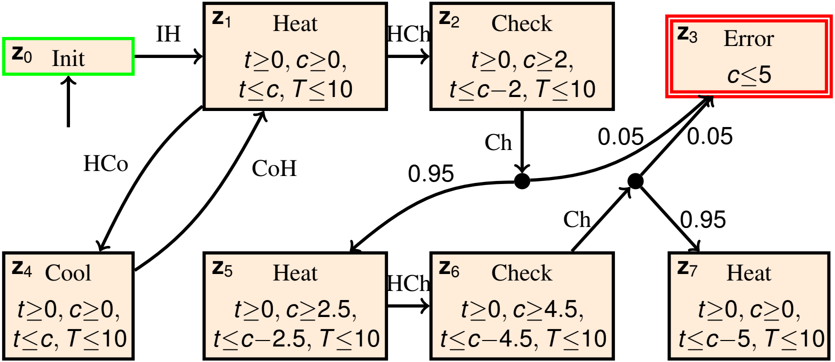


Abstraction of Probabilistic Hybrid Automata

PA $\mathcal{M} = (\mathbf{A}, \bar{\mathbf{z}}, \{\tau\} \uplus \mathit{Cmds}, \mathcal{T}_{\text{abs}})$

- $\mathbf{A}, \bar{\mathbf{z}}, \tau, \mathit{Cmds}$: as before
- $\mathcal{T}_{\text{abs}}$: **probabilistic**

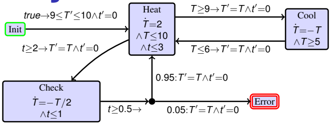


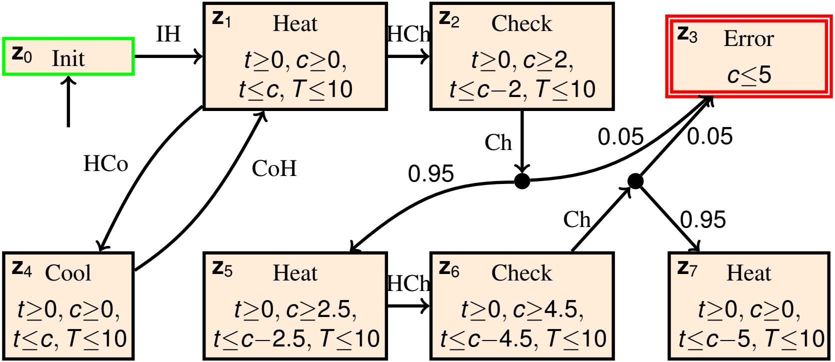


Abstraction of Probabilistic Hybrid Automata

PA $\mathcal{M} = (\mathbf{A}, \bar{\mathbf{z}}, \{\tau\} \uplus Cmds, \mathcal{T}_{\text{abs}})$

- $\mathbf{A}, \bar{\mathbf{z}}, \tau, Cmds$: as before
- $\mathcal{T}_{\text{abs}}$: **probabilistic**

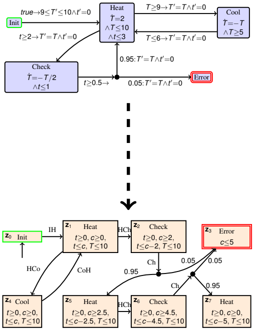




- how to obtain such an abstraction?

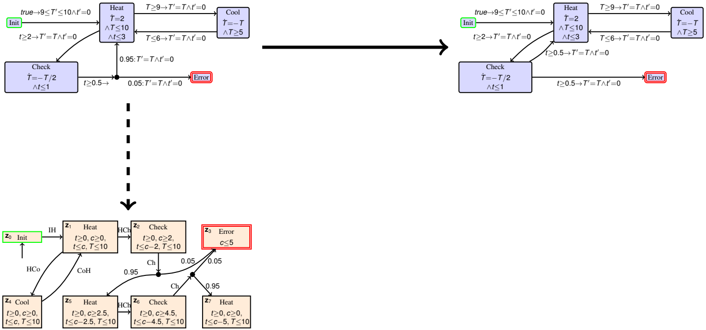
Constructing Abstractions of PHAs

- consider probabilistic hybrid automaton $\mathcal{H}$



Constructing Abstractions of PHAs

- consider probabilistic hybrid automaton $\mathcal{H}$
- consider non-probabilistic version $\mathrm{ind}(\mathcal{H})$ of $\mathcal{H}$
  replace $c = g \to p_1 : u_1 + \ldots + p_n : u_n$
  by $\mathrm{ind}(c) = \{g \xrightarrow{\ell_1} u_1, \ldots, g \xrightarrow{\ell_n} u_n\}$

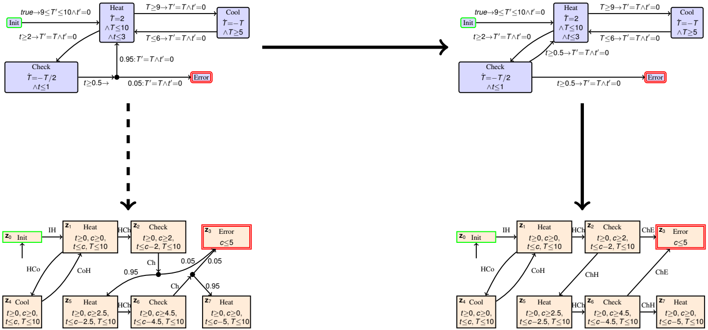


Constructing Abstractions of PHAs

- consider probabilistic hybrid automaton $\mathcal{H}$
- consider non-probabilistic version $\mathrm{ind}(\mathcal{H})$ of $\mathcal{H}$

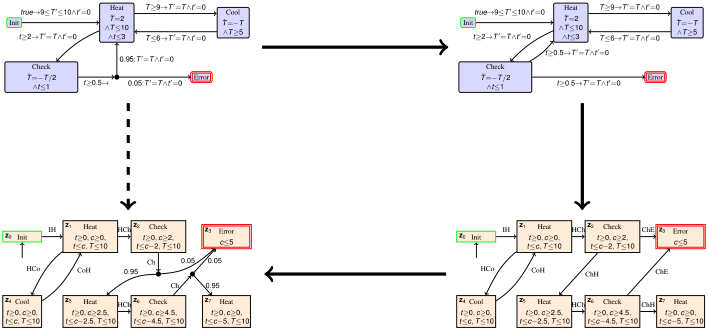
  replace $c = g \to p_1 : u_1 + \ldots + p_n : u_n$

  by $\mathrm{ind}(c) = \{g \xrightarrow{\ell_1} u_1, \ldots, g \xrightarrow{\ell_n} u_n\}$
- consider abstraction $\mathrm{abs}(\mathrm{ind}(\mathcal{H}))$ of $\mathrm{ind}(\mathcal{H})$



Constructing Abstractions of PHAs

- consider probabilistic hybrid automaton $\mathcal{H}$
- consider non-probabilistic version $\mathrm{ind}(\mathcal{H})$ of $\mathcal{H}$
  replace $c = g \to p_1 : u_1 + \ldots + p_n : u_n$
  by $\mathrm{ind}(c) = \{g \xrightarrow{\ell_1} u_1, \ldots, g \xrightarrow{\ell_n} u_n\}$
- consider abstraction $\mathrm{abs}(\mathrm{ind}(\mathcal{H}))$ of $\mathrm{ind}(\mathcal{H})$
- use $\mathrm{abs}(\mathrm{ind}(\mathcal{H}))$ to compute abstraction $\mathrm{abs}(\mathcal{H})$ of $\mathcal{H}$
  using labellings ℓ_i of $\mathrm{ind}(\mathit{Cmds})$

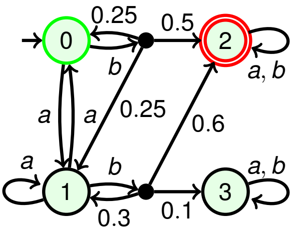


Correctness of PHA Abstraction

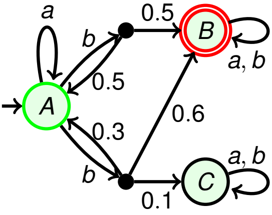
- follows from **simulation relation** semantics → abstraction
- $R \subseteq S \times \mathbf{A}$ [SegalaL95]
- $\overline{s}\, R\, \overline{\mathbf{z}}$

Correctness of PHA Abstraction

- follows from **simulation relation** semantics → abstraction
- $R \subseteq S \times \mathbf{A}$ [SegalaL95]
- $\overline{s}\, R\, \overline{\mathbf{z}}$
- if $s\, R\, \mathbf{z}$,
  for *a*-labelled successor **distribution** in simulated model,
  have *a*-labelled successor distribution in simulating model,
  so that the two are related

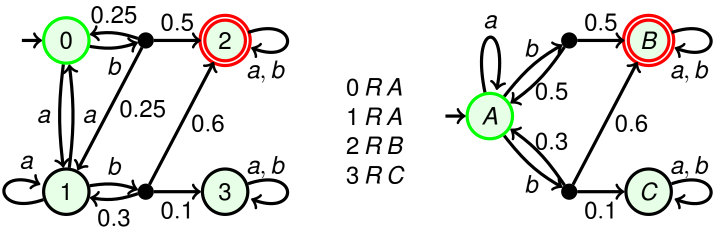


$0\, R\, A$
$1\, R\, A$
$2\, R\, B$
$3\, R\, C$



Correctness of PHA Abstraction

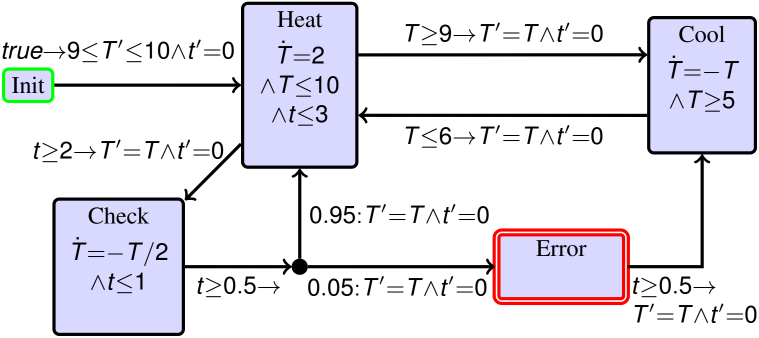
- follows from **simulation relation** semantics $\rightarrow$ abstraction
- $R \subseteq S \times \mathbf{A}$ [SegalaL95]
- $\overline{s}\, R\, \overline{\mathbf{z}}$
- if $s\, R\, \mathbf{z}$,
  for a-labelled successor **distribution** in simulated model,
  have a-labelled successor distribution in simulating model,
  so that the two are related



- maintains safeness

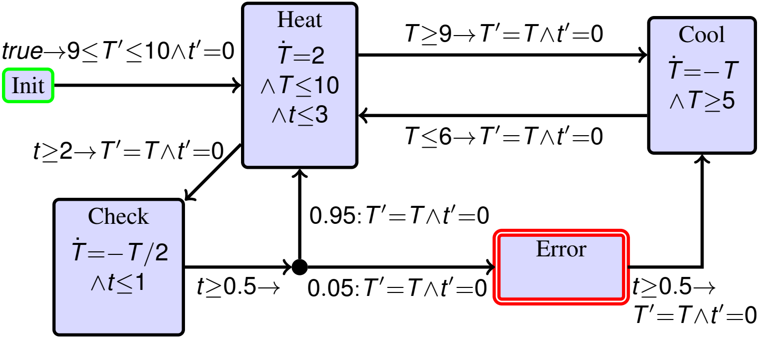
Rewards

- interest in properties other than reachability



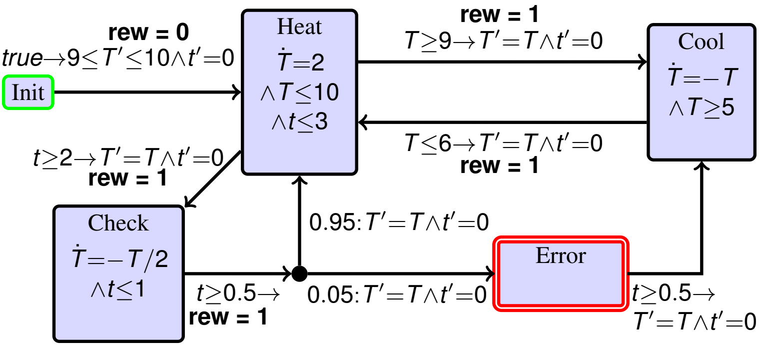
Rewards

- interest in properties other than reachability
- attach **rewards**



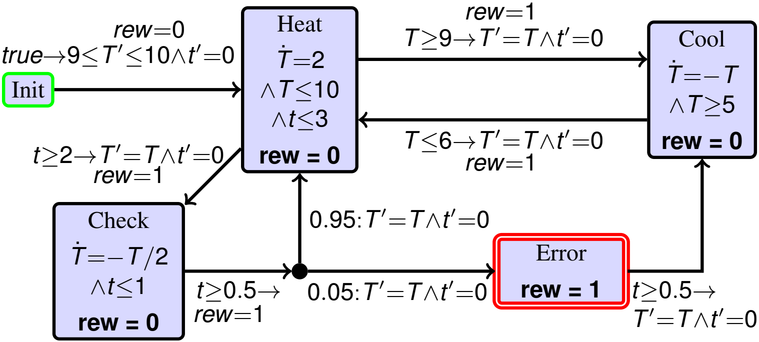
Rewards

- interest in properties other than reachability
- attach **rewards**
- obtained per command execution



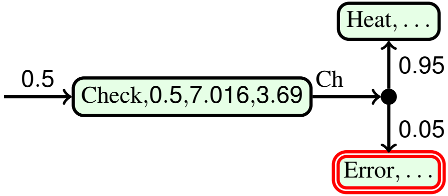
Rewards

- interest in properties other than reachability
- attach **rewards**
- obtained per command execution
- or per time unit



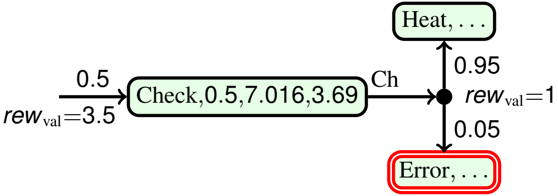
Rewards Semantics

- induce two reward structures in PA semantics



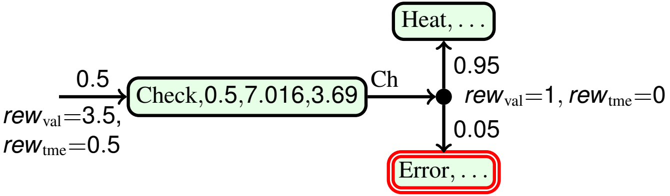
Rewards Semantics

- induce two reward structures in PA semantics
  value (rew_{val})



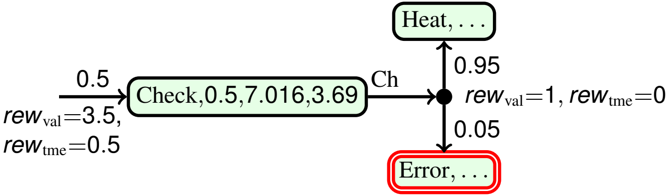
Rewards Semantics

- induce two reward structures in PA semantics
  value (rew_{val}) and time (rew_{tme})



Rewards Semantics

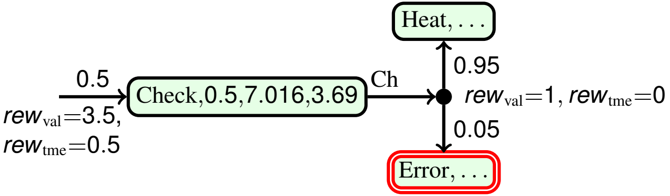
- induce two reward structures in PA semantics
  value ($\mathit{rew}_{\mathrm{val}}$) and time ($\mathit{rew}_{\mathrm{tme}}$)



- can now express properties
  accumulated: $\mathrm{val}^{\sigma}_{\mathcal{M},\mathit{rew},\mathrm{acc}} \stackrel{\mathrm{def}}{=} E_{\mathcal{M},\sigma}\left[\lim_{n\to\infty} \sum_{i=0}^{n} \mathit{rew}_{\mathrm{val}}\right]$
  long-run: $\mathrm{val}^{\sigma}_{\mathcal{M},\mathit{rew},\mathrm{lra}} \stackrel{\mathrm{def}}{=} E_{\mathcal{M},\sigma}\left[\lim_{n\to\infty} \frac{\sum_{i=0}^{n} \mathit{rew}_{\mathrm{val}}}{\sum_{i=0}^{n} \mathit{rew}_{\mathrm{tme}}}\right]$

Rewards Semantics

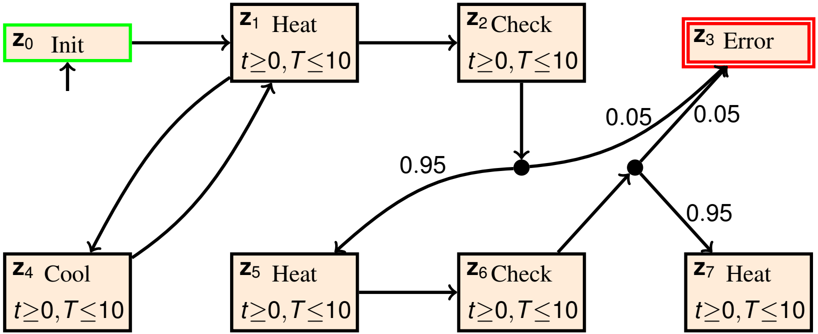
- induce two reward structures in PA semantics value (rew_{val}) and time (rew_{tme})



- can now express properties
  accumulated: $\mathrm{val}^{\sigma}_{\mathcal{M},rew,\mathrm{acc}} \stackrel{\mathrm{def}}{=} E_{\mathcal{M},\sigma}\left[\lim_{n\to\infty}\sum_{i=0}^{n} rew_{\mathrm{val}}\right]$
  long-run: $\mathrm{val}^{\sigma}_{\mathcal{M},rew,\mathrm{lra}} \stackrel{\mathrm{def}}{=} E_{\mathcal{M},\sigma}\left[\lim_{n\to\infty}\frac{\sum_{i=0}^{n} rew_{\mathrm{val}}}{\sum_{i=0}^{n} rew_{\mathrm{tme}}}\right]$
- interested in min/max of values

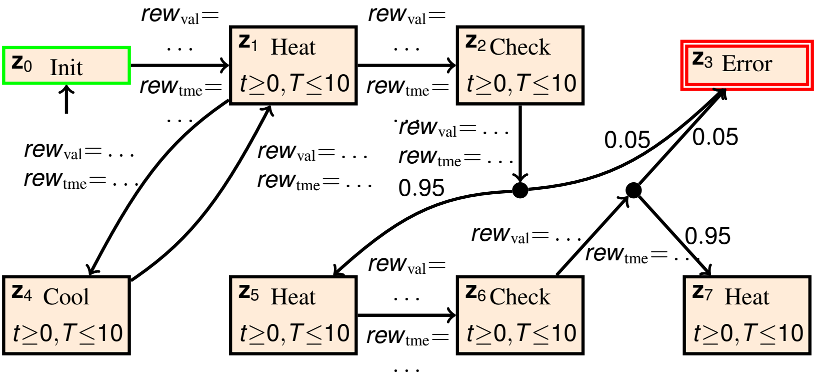
Abstraction for Rewards

- transform timed rewards → command rewards



Abstraction for Rewards

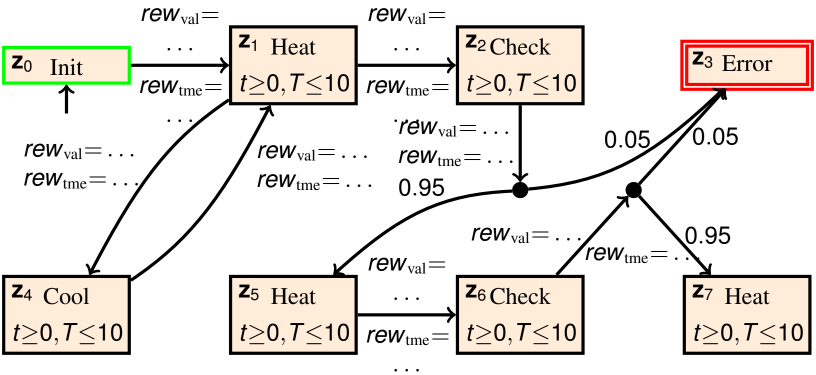
- transform timed rewards → command rewards
- compute reward structures for abstraction



Abstraction for Rewards

- transform timed rewards $\rightarrow$ command rewards
- compute reward structures for abstraction
- compute overapproximation of values

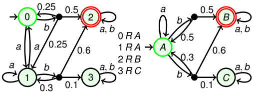
$\sigma :=$ initial scheduler
repeat
  compute reachability probability v under σ
  forall the z $\in$ **A do**
    improve $\sigma(\mathbf{z})$ if possible
until no further improvement possible
return (v, σ)

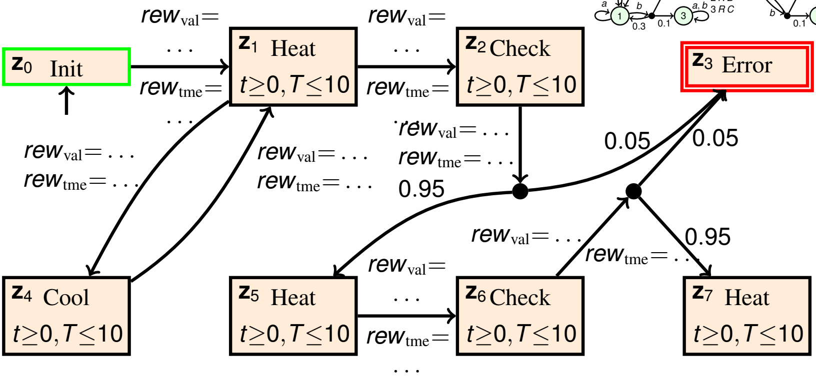


Abstraction for Rewards

- transform timed rewards → command rewards
- compute reward structures for abstraction
- compute overapproximation of values
- correctness: extended simulation relation

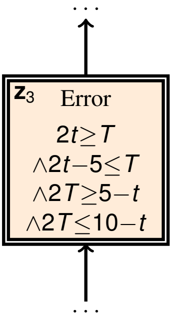
```
σ := initial scheduler
repeat
    compute reachability probability v under σ
    forall the z ∈ A do
        improve σ(z) if possible
until no further improvement possible
return (v, σ)
```





Computing Reward Structures

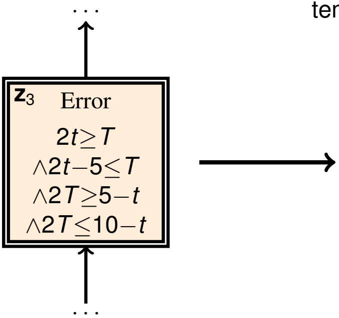
- depends on hybrid automata tool



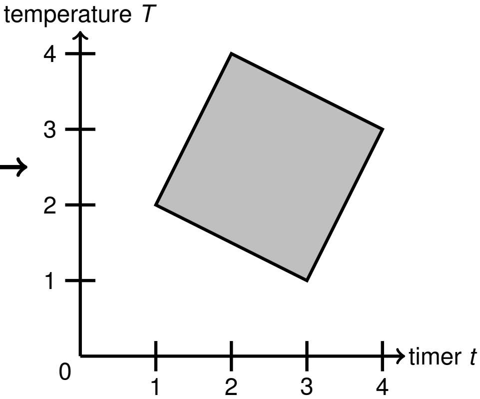
max time stay in error?

Computing Reward Structures

- depends on hybrid automata tool
- for polyhedra: maps to linear programming

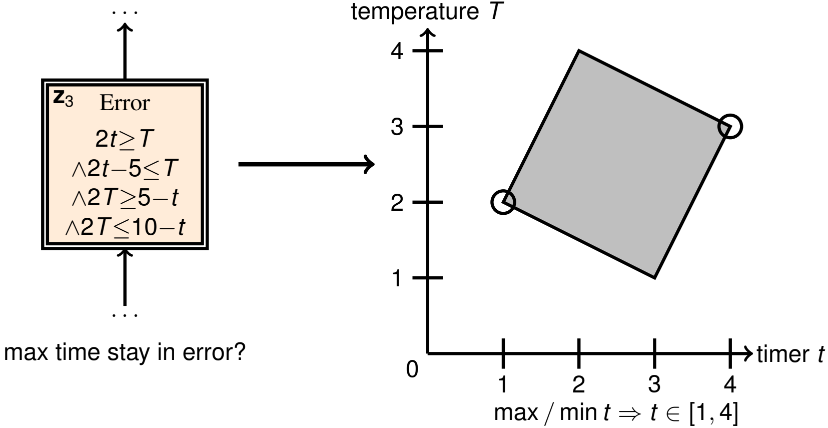


max time stay in error?



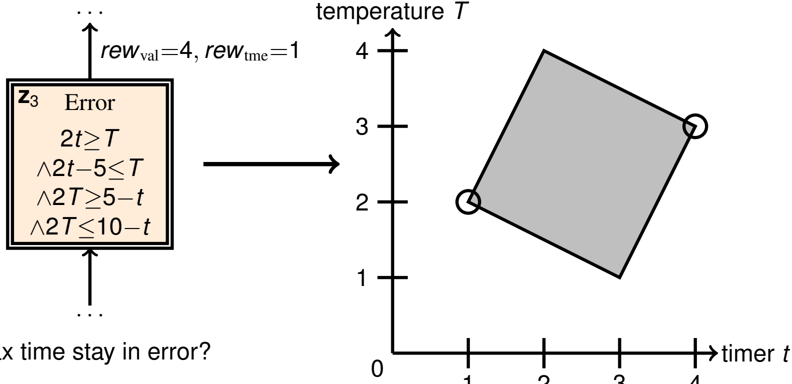
Computing Reward Structures

- depends on hybrid automata tool
- for polyhedra: maps to linear programming



Computing Reward Structures

- depends on hybrid automata tool
- for polyhedra: maps to linear programming



max time stay in error?

max / min $t \Rightarrow t \in [1, 4]$

Thesis Statement

Abstraction enables automatic verification of a very general class of properties of generic stochastic hybrid automata, by extending existing established methods.